VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Nástroje pro správu XML databáze

Management Tools for XML Database

2015 Daniel Zdražila

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Daniel Zdražila**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Nástroje pro správu XML databáze
Management Tools for XML Database

Zásady pro vypracování:

Cílem práce je navrhnout a vytvořit aplikaci s grafickým uživatelským rozhraním pro správu nativní XML databáze. Fyzické sestavování a vykonávání XQuery dotazů bude zajištěno některým z dostupných XQuery procesorů.

Práce bude splňovat následující body:
1. Popis jazyka XML, dotazovacího jazyka XQuery a současných nativních XML databází.
2. Modulární návrh architektury aplikace.
3. Návrh a implementace modulu pro zadávání dotazů XQuery a prezentaci výsledku.
4. Návrh a implementace modulu pro grafickou vizualizaci plánu dotazu.
5. Srovnání s existujícími aplikacemi.

Seznam doporučené odborné literatury:

Dle pokynů vedoucího.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Petr Lukáš**

Datum zadání: 01.09.2013
Datum odevzdání: 07.05.2015

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2015

Daniel Zdražila

.....

Chtěl bych tímto poděkovat svému vedoucímu bakalářské práce, panu Ing. Petru Lukášovi za průběžnou ochotu, odborné vedení a za čas, který věnoval při tvorbě práce.

Abstrakt

Tato bakalářská práce popisuje návrh a tvorbu aplikace s grafickým uživatelským rozhraním pro správu nativní XML databáze. V teoretické části je základní popis jazyka XML, dotazovacího jazyka XQuery a současných nativních XML databází. Dále je zde popsán modulární návrh architektury aplikace, návrh a implementace modulu pro zadávání dotazů XQuery a prezentaci výsledku a návrh a implementace modulu pro grafickou vizualizaci plánu dotazu. V závěru práce je srovnání s existujícími aplikacemi.

Klíčová slova: XML, XQuery, databáze, nativní XML databáze, bakalářská práce

Abstract

This bachelor thesis describes the design and development of the application with a graphical user interface for managing native XML databases. In the theoretical part there is a basic description of the XML language, XQuery query language and contemporary native XML databases. Furthermore, the modular design of application's architecture, design and implementation of the module for XQuery query and presentation of results and design and implementation of the module for graphical visualization and query plan. In the final chapter there is comparision with existing applications.

Keywords: XML, XQuery, database, native XML database, bachelor thesis

Seznam použitých zkratek a symbolů

SGML	–	Standard Generalized Markup Language
SQL	–	Structured Query Language
URI	–	Uniform Resource Identifier
W3C	–	World Wide Web Consortium
XML	–	eXtensible Markup Language
XPath	–	XML Path Language
XQuery	–	XML Query

Obsah

Seznam obrázků

1 Úvod

Jelikož jsou nativní XML databáze stále populárnější, jako téma bakalářské práce bylo zvoleno navržení a vytvoření vlastní aplikace s grafickým uživatelským rozhraním pro správu nativní XML databáze.

V teoretické části jsou popsány základy jazyka *XML*, dotazovacího jazyka *XQuery* a současných *nativních XML databází*.

Samotná aplikace je jednoduchá a uživatelsky příjemná. Pro fyzické sestavování a vykonávání XQuery dotazů jsou využity dostupné XQuery procesory, konkrétně *BaseX* a *RadegastXDB*. V této práci je postupně popsán modulární návrh architektury aplikace, návrh a implementace modulu pro zadávání dotazů XQuery a prezentaci výsledku a návrh a implementace modulu pro grafickou vizualizaci plánu dotazu. V závěru práce je srovnání s existujícími aplikacemi.

2 Teoretická část

Tato úvodní kapitola se věnuje základům značkovacího jazyka XML, dotazovacího jazyka XQuery a současným nativním XML databázím.

2.1 XML

Tato část o XML čerpá z [1], [7], [8], [9], [11], [10], [13], [4], [6] a [15].

XML neboli *eXtensible Marup Language* si můžeme představit, jako obecný značkovací jazyk, navržený a standardizovaný mezinárodním konsorciem *W3C*[1]. Standard XML vychází z původního, mnohem složitějšího a komplexnějšího jazyka *SGML*[2], je tudíž jeho podmnožinou a rozšířil se právě pro svou jednoduchost a zpracovatelnost.

Hlavní účel jazyka XML spočívá v ukládání informací v přehledné strukturované podobě. Díky tomu je možné efektivně vyhledávat a zpracovávat data. Jazyk XML je psán v textové, nikoliv binární podobě, člověk je tedy schopen v krajním případě data přečíst bez nutnosti využití aplikace, která by tato data zpracovala. Tento jazyk je také tzv. *case-sensitive*, což znamená, že se při psaní rozlišují velká a malá písmena. Nespornou výhodou je jeho rozšiřitelnost a nezávislost na platformách, tudíž je zajištěna jeho přenositelnost.

2.1.1 Element

Základním prvkem každého XML dokumentu je **element**. Elementy je možné do sebe navzájem vnořovat a tím vytvořit strukturovanou podobu celého XML dokumentu. V textu jsou elementy značeny pomocí tzv. *tagů*. Většina elementů se skládá z počátečního a koncového tagu. Tagy jsou ohraničeny znaky „<“ a „>“. Pro snadnou odlišitelnost od počátečního tagu, má koncový tag před svým názvem znak „/“.

Příklad elementu *<kniha>Zde se nachází obsah elementu kniha</kniha>*

V příkladu vidíme počáteční tag *<kniha>* a koncový tag *</kniha>*.

Může ovšem nastat případ, kdy element nemá žádný obsah, v tomto případě hovoříme o tzv. *prázdném elementu*. Prázdný element se dá zapsat dvěma způsoby:

1. *<kniha></kniha>*
2. *<kniha />*

2.1.2 Atribut

Atributy jsou dalšími prvky XML dokumentu. Jsou vkládány do počátečního tagu, jako jeho součást a upřesňují další informace o daném elementu. Atribut se skládá z *názvu atributu*, znaku „=“ a *hodnoty atributu*, kde hodnota atributu musí být uzavřena do uvozovek nebo apostrofů. Nutno podotknout, že každý element může, ale nemusí obsahovat jeden, či více atributů.

[1] W3C (World Wide Web Consortium) – `http://www.w3.org`.

[2] SGML (Standard Generalized Markup Language) – `http://www.w3.org/TR/REC-html40/intro/sgmltut.html`.

Příklad atributu: *<kniha žánr="Fantasy"> Zde se nachází obsah elementu kniha</kniha>*

V příkladu vidíme atribut *žánr* s hodnou *"Fantasy"*, jako součást elementu *kniha*.

2.1.3 Komentář

V XML dokumentu můžeme také samozřejmě vytvářet **komentáře**. Komentáře nám slouží jako poznámky, ale také ke zneviditelnění nechtěného obsahu. Pokud chceme text zakomentovat, zapíšeme ho mezi znaky „<!−−“ a „−−>“, přičemž platí, že komentář může obsahovat jakýkoliv text, kromě posloupnosti znaků „−−“.

Příklad komentáře: *<!−− zde je zakomentovaný text −−>*

2.1.4 Procesní istrukce

Procesní instrukce (PI) umožňují aplikacím zpracovávat XML. Zapisují se mezi znaky „<?“ a „?>“. Každá PI má svůj název a tvoří ji pouze atributy.

Příklad PI pro deklaraci XML dokumentu: *<?xml version="1.0"encoding="UTF-8"?>*

2.1.5 Správně strukturovaný (well-formed) XML dokument

Správně strukturovaný (well-formed) XML dokument je dokument, splňující následující pravidla:

1. obsahuje správnou XML deklaraci,
   tzn. použití procesní instrukce 2.1.4 *<?xml version="1.0"encoding="UTF-8"?>*[3],
2. XML dokument obsahuje právě jeden kořenový element,
3. pro každý počáteční tag, musí existovat odpovídající koncový tag, případně je počáteční tak zapsán, jako prázdný element,
4. nesmí docházet ke křížení elementů v dokumentu.

[3] Číslo verze a kódování se může měnit.

Obrázek 1: Příklad správně strukturovaného (well-formed) XML dokumentu. Na tomto dokumentu *Knihovna.xml* budeme ukazovat vlastnosti jazyka XQuery.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--zde je kořenový element Knihovna-->
<Knihovna>
  <kniha id="1">
    <nazev>Smrt krásných srnců</nazev>
    <autor>
      <jmeno>Ota</jmeno>
      <prijmeni>Pavel</prijmeni>
    </autor>
    <rok_vydani>1971</rok_vydani>
  </kniha>
  <kniha id="2">
    <nazev>Babička</nazev>
    <autor>
      <jmeno>Božena</jmeno>
      <prijmeni>Němcová</prijmeni>
    </autor>
    <rok_vydani>1855</rok_vydani>
  </kniha>
  <kniha id="3">
    <nazev>O myších a lidech</nazev>
    <autor>
      <jmeno>John</jmeno>
      <prijmeni>Steinbeck</prijmeni>
    </autor>
    <rok_vydani>1937</rok_vydani>
  </kniha>
</Knihovna>
```

2.2 XQuery

Tato část o XQuery čerpá z [2], [3], [5], [7], [12], [13] a [14].

XQuery, neboli *XML Query* je dotazovací jazyk navržený a standardizovaný mezinárodním konsorciem *W3C*. Je speciálně vytvořený pro dotazování nad XML dokumenty. Plní stejnou funkci, jako dotazovací jazyk *SQL*[4] v relačních databázích. To znamená, že v XML dokumentu umožňuje vyhledávat data, třídit data, prezentovat a formátovat výsledky. Díky těmto vlastnostem se stal v podstatě standardním dotazovacím jazykem v *nativních XML databázích*, viz 2.3.

XQuery je rozšířením jazyka XPath 2.0[5], který je jeho podmnožinou. Jelikož jazyk XQuery vychází z jazyka XPath 2.0, jsou všechny výrazy a funkce z XPath 2.0 obsažené také v XQuery. Ačkoliv je jazyk XPath 2.0 určený také pro dotazování nad XML dokumenty, je oproti XQuery velmi jednoduchý. Důvodem vytvoření XQuery bylo právě rozšíření funkcionality tohoto jazyka. Výhodou XQuery oproti XPath 2.0 je zejména možnost využití FLWOR výrazů, viz 2.2.3.

XQuery nejčastěji operuje nad sekvencemi. Může jít o sekvence XML uzlů nebo atomických hodnot (například celých čísel) – zápis (1, 2, 3) vytváří sekvenci.

2.2.1 Osy XPath

Osy XPath nám umožňují prohledávat části XML dokumentu vzhledem ke *kontextovému uzlu* (KU), tj. uzlu, na kterém se aktuálně nacházíme. Mezi nejpoužívanější osy patří *child* (přímí potomci KU), *descendant-or-self* (KU a všichni potomci) a *attribute* (atrubuty KU). Kompletní popis všech os lze najít na [3].

Příklad XPath dotazu nad ukázkovým dokumentem *Knihovna.xml*, viz obrázek1:

```
/Knihovna//kniha[@id="1"]/nazev
```

Ze zdrojového dokumentu vybereme kořenový element pomocí `/Knihovna`.
Následuje `//kniha` – vyhledá všechny potomky s názven „kniha“. Predikát `[@id="1"]` vyfiltruje všechny uzly – knihy s hodnotou atrbuti `id` rovnou „1“. Ze vzniklé množiny jsou pak vybráni všichni přímí potomci s názvem „nazev“.

Výsledek dotazu: *<nazev>Smrt krásných srnců</nazev>*

Uvedený příklad byl zjednodušením dotazu:

```
/child::Knihovna//descendant-or-self::node()/kniha
    [attribute::id="1"]/child::nazev
```

2.2.2 Funkce v XQuery

XQuery obsahuje celou řadu vestavěných funkcí, které nám pomáhají při práci s daty. Kompletní popis všech funkcí není předmětem této práce a lze jej najít na [2]. Zde je pro ukázku popis několika důležitých.

[4]SQL (Structured Query Language) – Dotazovací jazyk pro relační databáze.
[5]XPath 2.0 (XML Path Language) – `http://www.w3.org/TR/xpath20/`.

Funkce *doc()* umožňuje čtení dat přímo ze souboru. Má jeden vstupní parametr – URI[6] dokumentu. Vrací celý obsah zadaného dokumentu.

doc('Knihovna.xml') vrátí celý obsah dokumentu *Knihovna.xml*.

Funkce *collection()* umožňuje čtení dat i z několika souborů. Má jeden vstupní parametr – URI kolekce. Vrací celý obsah všech dokumentů v kolekci.

XML databáze se obvykle skládá z několika kolekcí a kolekce se skládají z několika dokumentů. Představme si XML databázi obsahující kolekci s názvem *Knihovna*, ve které jsou uloženy dva dokumenty, první s názvem *Knihy.xml* a druhý s názvem *Časopisy.xml*.

collection('Knihovna') vrátí celý obsah obou dokumentů[7].

Agregační funkce (*count()*, *sum()*, *avg()*, *min()*, *max()*)

- U agregačních funkcí jsou vstupními parametry sekvence
- **count()** – vrací počet položek v sekvenci
- **sum()** – vrací součet hodnot v sekvenci
- **avg()** – vrací průměrnou hodnotu v sekvenci
- **min()** – vrací minimální hodnotu v sekvenci
- **max()** – vrací maximální hodnotu v sekvenci

Funkce *count((1,1,1))* vrátí výsledek 3.

2.2.3 FLWOR výrazy

Největší sílu a takovou oblíbenost získalo XQuery právě díky **FLWOR** výrazům [čteme jako flower].

Název FLWOR je složený z prvních písmen jednotlivých *klauzulí*, kde těmito klauzulemi jsou **for**, **let**, **where**, **order by** a **return**. Svou strukturou a zápisem připomínají FLWOR výrazy kombinaci *foreach* cyklu jazyka *C#*[8] a *SELECT*, *FROM*, *WHERE*, *ORDER BY* jazyka *SQL*.

XQuery je *deklarativní jazyk*, tzn. dotazem říkáme, co chceme na výstupu, ale nespecifikujeme jak. To znamená, že sice píšeme **for**, které můžou připomínat *foreach*, ale o skutečném algoritmu zpracování dotazu rozhodne procesor dotazu.

Klauzule *for* slouží pro vývěr posloupnosti uzlů, s nimiž se pracuje. Je povinná, pokud není zavedena klauzule *let* nebo jiná klauzule *for*. Klauzule **for** vypadá tak, že napíšeme „for“, následuje proměnná (znak $ a název proměnné), klíčové slovo „in“ a specifikace sekvence.

[6] URI (Uniform Resource Identifier) – je textový řetězec s definovanou strukturou, který slouží k přesné specifikaci zdroje informací. Zdroj: `http://cs.wikipedia.org/wiki/Uniform_Resource_Identifier`.

[7] Obvykle však nepožadujeme výpis celého dokumentu, takže na funkci navazuje použití nějakého XPath dotazu, který vyhodnotí nějaký výsledek.

[8] Jazyk C# je objektově orientovaný programovací jazyk, vytvořený počítačovou firmou Microsoft.

Klauzule *let* slouží pro přiřazení výsledků do proměnné. Je povinná, pokud není zavedena klauzule *for* nebo jiná klauzule *let*. Klauzule **let** vypadá tak, že napíšeme „let“, následuje proměnná (znak $ a název proměnné), poté „:=“ a nakonec hodnota, kterou chceme přiřadit.

Klauzule *where* slouží, jako omezující podmínka. Je nepovinná a obsahuje výraz, jehož výsledkem je boolovská hodnota, která filtruje výstup.

Klauzule *order by* slouží pro seřazení výsledků podle kritéria. Je nepovinná.

Klauzule *return* slouží pro vrácení výsledku. Je povinná.

Příklad FLWOR výrazu nad ukázkovým dokumentem *Knihovna.xml*, obr. 1

```
for $x in doc('Knihovna.xml')//kniha
let $y := $x/rok_vydani
where $y>1900
order by $x/nazev
return $x/nazev
```

Výsledek dotazu:

```
<nazev>O myších a lidech</nazev>
<nazev>Smrt krásných srnců</nazev>
```

Pomocí XQuery máme možnost vytvářet vlastí XML uzly. Tzn. XQuery může sloužit i k naformátování výsledku a neslouží jen k filtrování dokumentu.

Pokud například budeme chtít výsledek výše zmíněného FLWOR dotazu vložit, jako obsah nového XML uzlu, můžeme to provést následujícím způsobem:

```
<nazvy_knih>
{
for $x in doc('Knihovna.xml')//kniha
let $y := $x/rok_vydani
where $y>1900
order by $x/nazev
return $x/nazev
}
</nazvy_knih>
```

Výsledek dotazu pak bude vypadat:

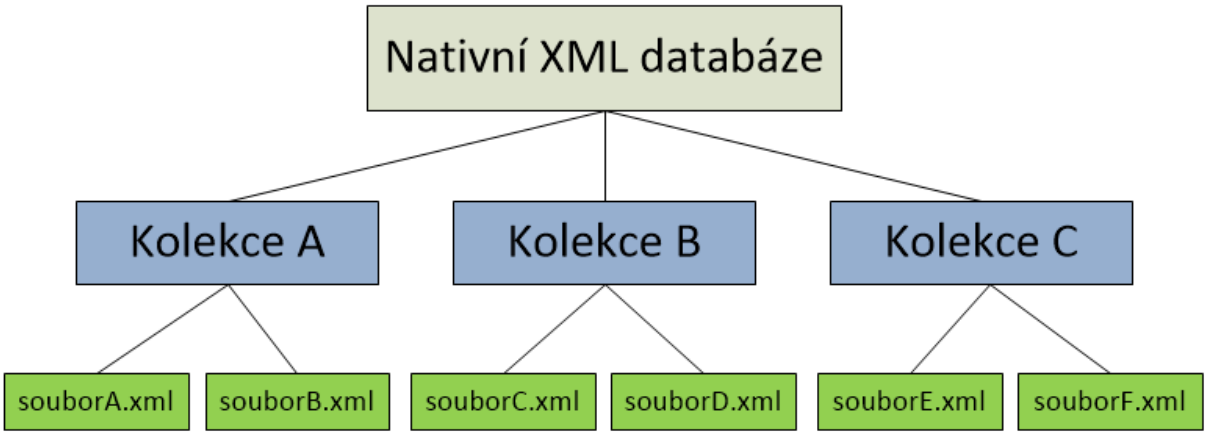
```
<nazvy_knih>
   <nazev>O myších a lidech</nazev>
   <nazev>Smrt krásných srnců</nazev>
</nazvy_knih>
```

2.3 Nativní XML databáze

Tato část o nativních XML databázích čerpá z [8], [12], [13], [16], [22] a [23].

Pod pojmem XML si obvykle představíme textový soubor obsahující přímo čitelný XML dokument. Pro efektivní zpracování XML ale obvykle není textová podoba dokumentu vhodná, zejména pokud jde o velké XML dokumenty a kolekce čítající řádově např. milióny elementů. Z toho důvodu vznikají tzv. *nativní XML databáze*, ve kterých se XML ukládá pomocí persistentních datových struktur jako jsou heap tabulka nebo B-strom [24]. Tyto struktury pak umožňují efektivní dotazování a to i přes případnou velikost kolekce. Pro dotazování nad nativními XML databázemi se používají nejčastěji jazyky *XQuery* a *XPath*, viz 2.2. Obvykle se nativní databáze skládá z několika kolekcí, každá kolekce obsahuje XML soubory, viz obrázek 2.

Mezi současné nativní XML databáze můžeme zařadit například *BaseX* [17], *MonetDB* [21], *eXistdb* [18], *MarkLogic* [19], *Qizx* [20] nebo *RadegastXDB* – prototyp nativní XML databaze vyvíjený VŠB – Technická univerzita Ostrava. Tato vytvořená aplikace s tímto prototypem může komunikovat.



Obrázek 2: Ukázkový strom uspořádání nativní XML databáze

2.3.1 Nativní XML databáze BaseX

BaseX je výkonný open source databázový systém napsaný v jazyce *Java*[9], který je vyvíjen pod licencí *BSD Licence*[10] a je nezávislý na platformách. Současná (květen, 2015) verze je BaseX 8.1.1. Součástí BaseX je vlastní *XQuery procesor* (BaseX XQuery Processor), *klient/-server architektura* (BaseX Client/Server Architecture) a nechybí ani *grafické uživatelské rozhranní* (BaseX Graphical User Interface (GUI)). Podporované dotazovací jazyky jsou XPath a XQuery.

[9]Jazyk Java je objektově orientovaný programovací jazyk, který vyvinula firma Sun Microsystems. Zdroj: `http://cs.wikipedia.org/wiki/Java_(programovací_jazyk)`

[10]BSD Licence – `http://cs.wikipedia.org/wiki/BSD_licence`

2.3.2 Nativní XML databáze MonetDB

MonetDB/XQuery je open source databázový systém, který obsahuje zcela kompletní podporu pro jazyk XQuery, včetně modulů, XQUF[11] a uživatelsky definované funkce. Systém poskytuje vysoký výkon a je škálovatelný pro velké XML kolekce.

Nutno podotknout, ze projekt *MonetDB/XQuery* byl zastaven v březnu 2011. Přesto je v současné době velmi používaný a patří mezi nejvýkonnější a nejrychlejší nativní XML databázové systémy.

2.3.3 Nativní XML databáze eXistdb

eXistdb je open source databáze, řadící se to tzv. *NoSQL*[12] databází. Je psána v jazyce *Java*. eXistdb je vyvíjena pod licencí *LGPL*[13]. Současná verze je eXistdb 2.2. Podporované dotazovací jazyky jsou XPath a XQuery.

2.3.4 Nativní XML databáze MarkLogic

MarkLogic je komerční databáze, řadící se do *NoSQL* databází.

MarkLogic je dostupná v několika variantách:

- **Developer** – Je zdarma, obsahuje všechny klíčové funkce MarkLogic
- **Essential Enterprise** – je placená, určená pro firemní nasazení
- **Global Enterprise** – je placení, určená pro globálně distribuované aplikace

2.3.5 Nativní XML databáze Qizx

Qizx je komerční databáze, řadí se do *NoSQL* databází. Je psána v jazyce *Java*. Obsahuje vlastní *Qizx Engine*, skládající se z *XQuery procesoru* (XQuery Processor), *kompileru* (Query Compiler) a *nativní XML databáze* (Native XML Database Storage & Index).

2.3.6 Nativní XML databáze RadegastXDB

V rámci databázové skupiny na VŠB – Technická univerzita Ostrava, katedra informatiky, vzniká prototyp nativní XML databaze **RadegastXDB**, který se specializuje na velké XML kolekce o velikosti i několika GB a je efektivní ve vykonávání strukturálních XPath dotazů. V tuto chvíli nepodporuje XQuery Update Facility.

[11] XQUF (XQuery Update Facility) – `http://www.w3.org/TR/xquery-update-10/`

[12] NoSQL je databázový koncept, ve kterém datové úložiště i zpracování dat používají jiné prostředky než tabulková schémata tradiční relační databáze. Zdroj: `http://cs.wikipedia.org/wiki/NoSQL`

[13] LGPL – `http://cs.wikipedia.org/wiki/GNU_Lesser_General_Public_License`

3 Modulární návrh architektury aplikace

3.1 Úvod

Celá aplikace je napsaná v programovacím jazyku *C#*, jako desktopová aplikace. Vývojovým prostředím bylo zvoleno *Microsoft Visual Studio Ultimate 2012*. Vizuální návrh aplikace a rozložení jednotlivých komponent je inspirováno *Microsoft SQL Server Management Studiem 2012*. Při implementaci byly použity volně dostupné externí knihovny *FastColoredTextBox.dll* a *WeifenLuo.WinFormsUI.Docking.dll*. Vykonávání XQuery dotazů je zajištěno XQuery procesorem *BaseX* (viz 2.3.1) ve verzi BaseX 7.9 a prototypem *RadegastXDB* (viz 2.3.6).

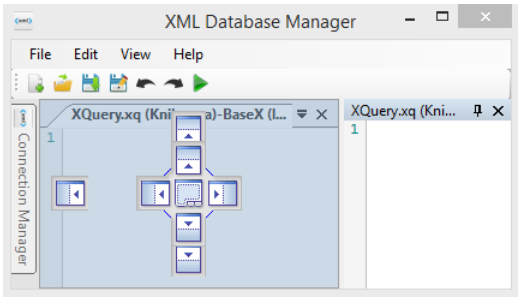
3.2 Použité komponenty

3.2.1 Fast Colored TextBox

Fast Colored TextBox[14] je komponenta určená k editaci textu a zvýrazňování syntaxe zdrojových kódů. Vytvořil ji Pavel Torgashov a je dostupná pod licencí *The GNU Lesser General Public License (LGPLv3)*. Svými funkcemi připomíná *RichTextBox*, oproti němu je ale rychlejší při zpracovávání velkého množství textu, nabízí již v základu číslování řádků, nastavování stylů písma, barvy pozadí, snadno získává přístup k textu pomocí regulárních výrazů, několika úrovňové Undo/Redo funkce a mnoho dalšího. Ve Visual Studiu lze tuto komponentu nainstalovat přes *NuGet Package Manager – Package Manager Console* příkazem `Install-Package FastColoredTextBox`. V aplikaci, byla tato komponenta použita pro zvýrazňování syntaxe XQuery.

3.2.2 Weifen Luo DockPanelSuite

Weifen Luo DockPanelSuite[15] je open source dokovací knihovna pro .NET Windows Forms určená pro práci s okny, vyvinul ji Weifen Luo. Mezi hlavní vlastnosti můžeme zařadit automatické schování okna, možnost plovoucího okna, vnořené dokování nebo drag-and-drop – umožňující uživateli přetáhnout a zachytit okno k libovolnému okraji havního okna, viz obrázek 3. Ve Visual Studiu lze tuto komponentu nainstalovat přes *NuGet Package Manager – Package Manager Console* příkazem `Install-Package DockPanelSuite`.



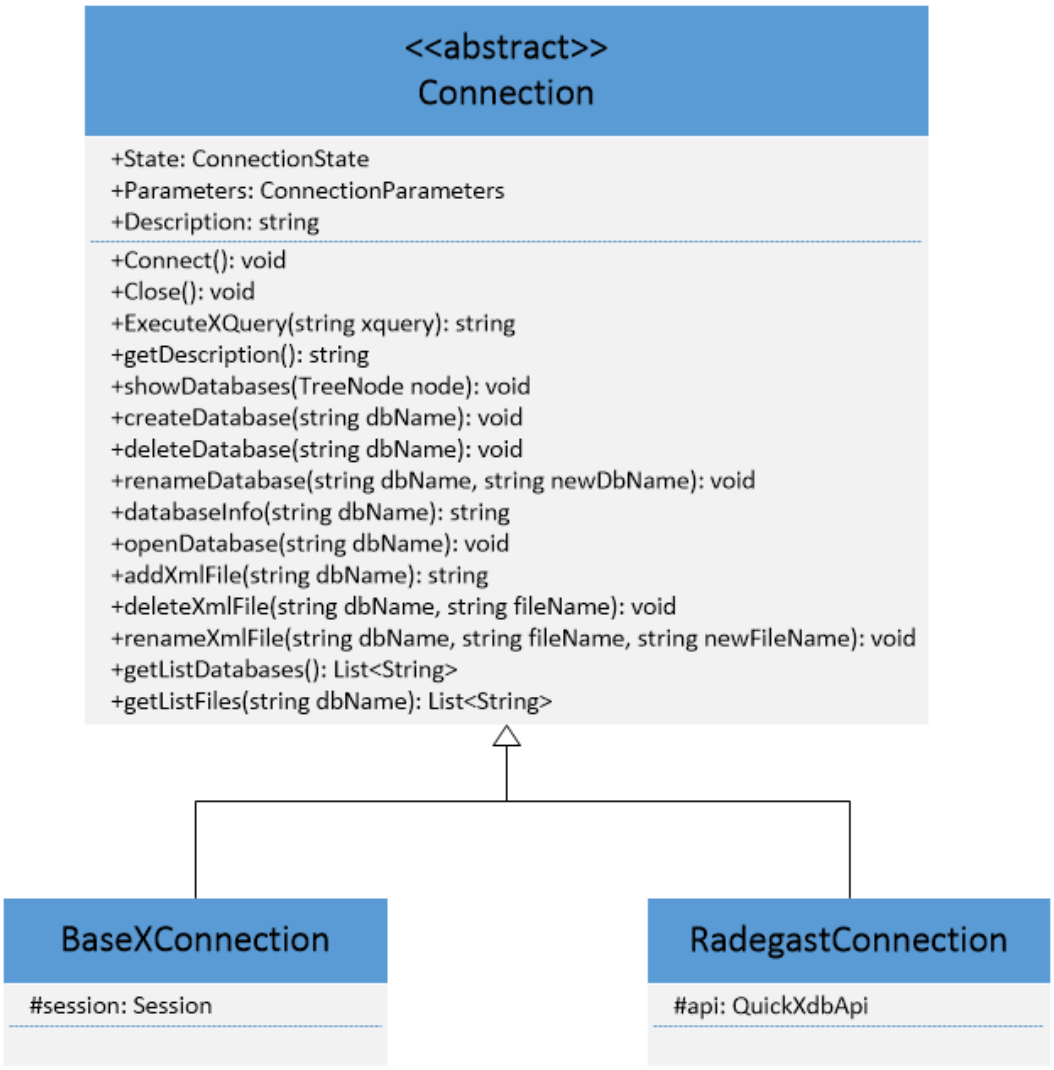
Obrázek 3: Ukázka vlastností komponenty Weifen Luo DockPanelSuite

[14]Pro více informací ohledně Fast Colored TextBoxu navštivte: `http://www.codeproject.com/Articles/161871/Fast-Colored-TextBox-for-syntax-highlighting`

[15]Pro více informací ohledně Weifen Luo DockPanelSuite navštivte: `http://www.websofia.com/2013/03/weifenluo-dockpanelsuite-tutorial-cookbook/`

3.3 Komunikace s databází

Aplikaci bylo žádoucí navrhnout tak, aby byla nezávislá na použité nativní XML databázi. Jelikož má každá databáze své specifické nároky na připojení a provádění XQuery dotazů, byla komunikace mezi jednotlivými databázemi navržena podle návrhového vzoru *abstraktní továrna*[16], kdy byla vytvořena abstraktní třída **Connection**, ve které jsou abstraktně deklarovány všechny potřebné metody. Každá databáze si pak jednotlivé metody implementuje podle své potřeby. Tento návrh je zobrazen na obrázku 4. Komunikace s konkrétními databázemi a významy metod jsou vysvětleny v následující části práce (viz 3.3.1 a 3.3.2).



Obrázek 4: Třídní diagram komunikace s databází

[16] Abstraktní továrna – `http://cs.wikipedia.org/wiki/Abstraktní_továrna`

3.3.1 Komunikace s BaseX

Pro navázání spojení a komunikaci s databází byly k dispozici zdrojové kódy (aplikační programové rozhraní), dostupné z oficiálních stránek BaseX [17]. Třída `Session` slouží pro navázání spojení s BaseX serverem prostřednictvím konstruktoru s parametry `host`, `port`, `username` a `pw`. Pro provedení dotazu je zde metoda `Execute()` se vstupním parametrem v podobě textového řetězce. Spojení se ukončuje metodou `Close()`. Nyní zde budou postupně popsány jednotlivé metody ve třídě `BaseXConnection`.

Connect() je metoda, která vytváří nové spojení s databází BaseX. Vytváří novou instranci třídy `Session` se vstupními parametry host, číslem portu, uživatelkým jménem a heslem.

Close () ukončuje stávající spojení voláním metody `Close()` na instanci třídy `Session`.

ExecuteXQuery(string xquery) posílá na server požadavek pomocí zadaného XQuery dotazu v podobě textového řetězce, jako parametr metody. Je zde volána metoda `Execute()` na instanci třídy `Session`. Server rozpozná, že se jedná o požadavek na vyhodnocení XQuery dotazu podle klíčového slova „`xquery`“, uvedeného na začátku dotazu. Metoda výsledek dotazu vrátí v podobě textového řetězce.

getDescription() vrátí popisek připojení v následující podobě:
`BaseX (`*`«hostname»`*`) port:` *`«číslo portu»`*.

getListDatabases() posílá na server požadavek na výpis všech databází pomocí klíčového slova „`List`“. Server požadavek zpracuje a vrátí výsledek v podobě textového řetězce, který má podobu tabulky se sloupci `Name`, `Resources`, `Size` a `Input Path`. Metoda vyparsuje z vráceného textového řetězce pouze názvy (sloupec `Name`) databází a vrátí je v podobě kolekce textových řetězců.

getListFiles(string dbName) posílá na server požadavek na výpis všech XML dokumentů uložených v konkrétní databázi, určené podle vstupního parametru `dbName`, pomocí klíčových slov „`List` *`«název databáze»`*“. Server požadavek zpracuje a vrátí výsledek v podobě textového řetězce, který má podobu tabulky se sloupci `Input Path`, `Type`, `Content-Type` a `Size`. Metoda vyparsuje z vráceného textového řetězce pouze názvy (sloupec `Input Path`) souborů a vrátí je v podobě kolekce textových řetězců.

showDatabases(TreeNode node) vykresluje stromovou strukturu v Connection Manageru. Vstupním parametrem je Uzel stromu (objekt `TreeNode`), do kterého se má naplnit seznam databází, jejich kolekcí a dokumentů.

createDatabase(string dbName) vytvoří novou databázi s názvem podle vstupního parametru `dbName`. Na server se pošle požadavek pomocí klíčových slov „`create db` *`«název databáze»`*“.

deleteDatabase(string dbName) smaže databázi s nazvem podle vstupního parametru `dbName`. Na server se pošle požadavek pomocí klíčových slov „`drop db «název databáze»`“.

renameDatabase(string dbName, string newDbName) přejmenuje aktuální databázi s nazvem podle vstupního parametru `dbName` na nový název pomocí `newDbName`. Na server se pošle požadavek pomocí klíčových slov „`alter db «starý název databáze» «nový název databáze»`“.

openDatabase(string dbName) otevírá databázi a umožňuje tím nad ní provádět dotazy. Jako vstupní parametr je název databáze `dbName`. Na server se pošle požadavek pomocí klíčových slov „`open «název databáze»`“.

databaseInfo(string dbName) vrací informace o databázi určené pomocí vstupního parametru `dbName` v podobě textového řetězce. V metodě je nejprve nutné otevřít databázi pomocí metody `openDatabase(string dbName)`. Na server se pak pošle požadavek pomocí klíčových slov „`info database`“.

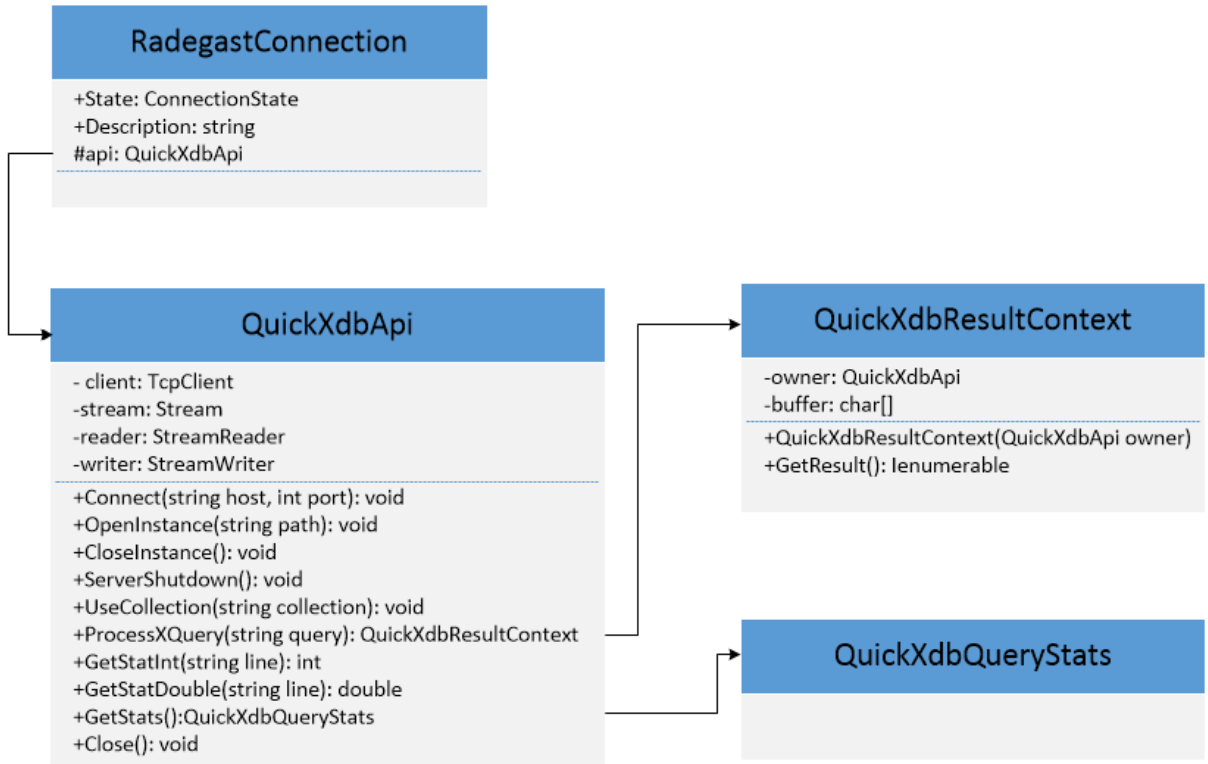
addXmlFile(string dbName) přidává XML dokument do databáze, určené pomocí vstupního parametru `dbName`. V metodě je nejprve nutné otevřít databázi pomocí metody `openDatabase(string dbName)`. Poté pomocí `Open File Dialogu` vyberu soubor, který chci přidat do databáze. Na server se pak pošle požadavek pomocí klíčových slov „`add «název souboru»`“.

deleteXmlFile(string dbName, string fileName) maže z databáze dané parametrem `dbName` XML dokument daný parametrem `fileName`. V metodě je nejprve nutné otevřít databázi pomocí metody `openDatabase(string dbName)`. Na server se pak pošle požadavek pomocí klíčových slov „`delete «název souboru»`“.

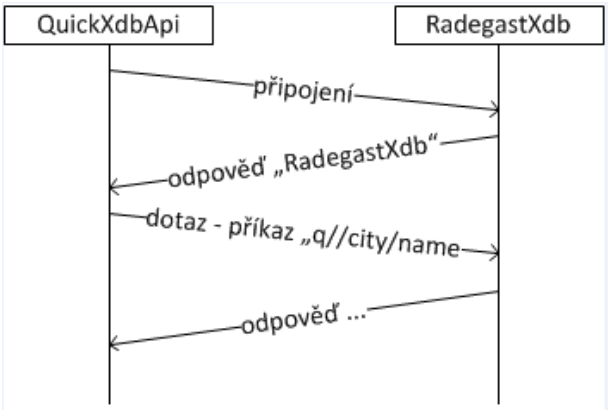
renameXmlFile(string dbName, string fileName, string newFileName) přejmenuje XML dokument s aktuálním názvem daný parametrem `fileName` v databázi dané parametrem `dbName` na nový název daný parametrem `newFileName`. V metodě je nejprve nutné otevřít databázi pomocí metody `openDatabase(string dbName)`. Na server se pak pošle požadavek pomocí klíčových slov „`rename «starý název souboru» «nový název souboru»`“.

3.3.2 Komunikace s RadegastXDB

Komunikace se serverem je zjištěna pomocí třídy `QuickXdbApi`, která je zobrazena na obrázku 5. Třída `QuickXdbApi` komunikuje se serverem RadegastXDB na základě protokolu `TCP/IP`. Příklad komunikace můžeme vidět na obrázku 6. Rozhraní `QuickXdbApi` vzniklo ještě před návrhem obecné struktury tříd pro připojení v této aplikaci. `QuickXdbApi` je obecné nezávislé API (aplikační programové rozhraní) pro praci s RadegastXDB a může se použít nezávisle v jiné aplikaci.



Obrázek 5: Třídní diagram komunikace s databází RadegastXDB



Obrázek 6: Ukázka komunikace mezi QuickXdbApi a RadegastXDB

Nyní zde budou postupně popsány jednotlivé metody ve třídě `QuickXdbApi`.

Connect(string host, int port) zahajuje spojení se serverem. Vstupními parametry jsou hostname a číslo portu. Spojení proběhne úspěšně, pokud odpověď serveru bude „`RadegastXDB`“.

OpenInstance(string path) otevírá instanci na serveru. Vstupní parametr je cesta k instanci. Na server se pošle požadavek pomocí klíčových slov „`OPEN «cesta k instanci»`“. Instance je otevřená, pokud server odpoví „`OK`“.

CloseInstance() zavírá instanci na serveru. Na server se pošle požadavek pomocí klíčového slova „`CLOSE`“. Instance se zavře, pokud server odpoví „`OK`“.

ServerShutdown() vypíná server. Na server se pošle požadavek pomocí klíčového slova „`EXIT`“. Server je vypnut, pokud je odpověď „`OK`“.

UseCollection(string collection) slouží pro nastavení aktuální kolekce na serveru. V RadegastXDB je vždy jedna kolekce aktuální a nad tou se spouští dotazy. Vstupním parametrem je název kolekce. Na server se pošle požadavek pomocí klíčových slov „`USE «název kolekce»`“. Kolekce se nastaví, pokud server odpoví „`OK`“.

ProcessXQuery(string query) posílá na server požadavek pomocí zadaného XQuery dotazu v podobě textového řetězce, jako parametr metody. Server rozpozná, že se jedná o požadavek na vyhodnocení XQuery dotazu, jestliže příkaz začíná písmenem „`Q`“, uvedeného před samotným XQuery dotazem. Pokud je odpověď serveru „`OK`“, metoda výsledek dotazu vrátí v podobě nové instance kontextu pro iteraci vysledkem `QuickXdbResultContext`.

Třída `QuickXdbResultContext` obsahuje metodu `GetResult()`, která vrací iterátor (`IEnumerable`) přes jednotlivé položky výsledku. Postupně se vyžaduje další a další výsledek. Dokud server odpovídá „`ITEM`“, znamená to, že následuje jedna položka výsledku. Pokud neodpoví „`ITEM`“, znamená to konec výsledku.

Nyní zde budou postupně popsány jednotlivé metody ve třídě `RadegastConnection`.

Connect() vytváří nové spojení s databází RadegastXDB. Vytváří novou instranci třídy `QuickXdbApi` se vstupními parametry `host` a `port`. Současně otevírá instanci, pomocí metody `OpenInstance(string path)` a nastavuje kolekci metodou `UseCollection(string collection)`. Kolekce se vždy jmenuje „`Collection`“.

Close () ukončuje stávající spojení voláním metody `Close()` na instanci třídy `QuickXdbApi`.

ExecuteXQuery(string xquery) je metoda, která posílá na server požadavek pomocí zadaného XQuery dotazu v podobě textového řetězce, jako parametr metody. Je zde volána metoda `ProcessXQuery(xquery)` z instance `QuickXdbApi`, která vzniká při otevření spojení. Metoda výsledek dotazu vrátí v podobě textového řetězce.

getDescription() vrátí popisek připojení v následující podobě: „`Pripojeni RadegastXDB`“.

getListDatabases() vrátí název databáze. Defaultně je název databáze nastaven na „`Default Database`“.

getListFiles(string dbName) vrátí název XML dokumentu. Defaultně je název dokumentu nastaven na „`Default XML`“.

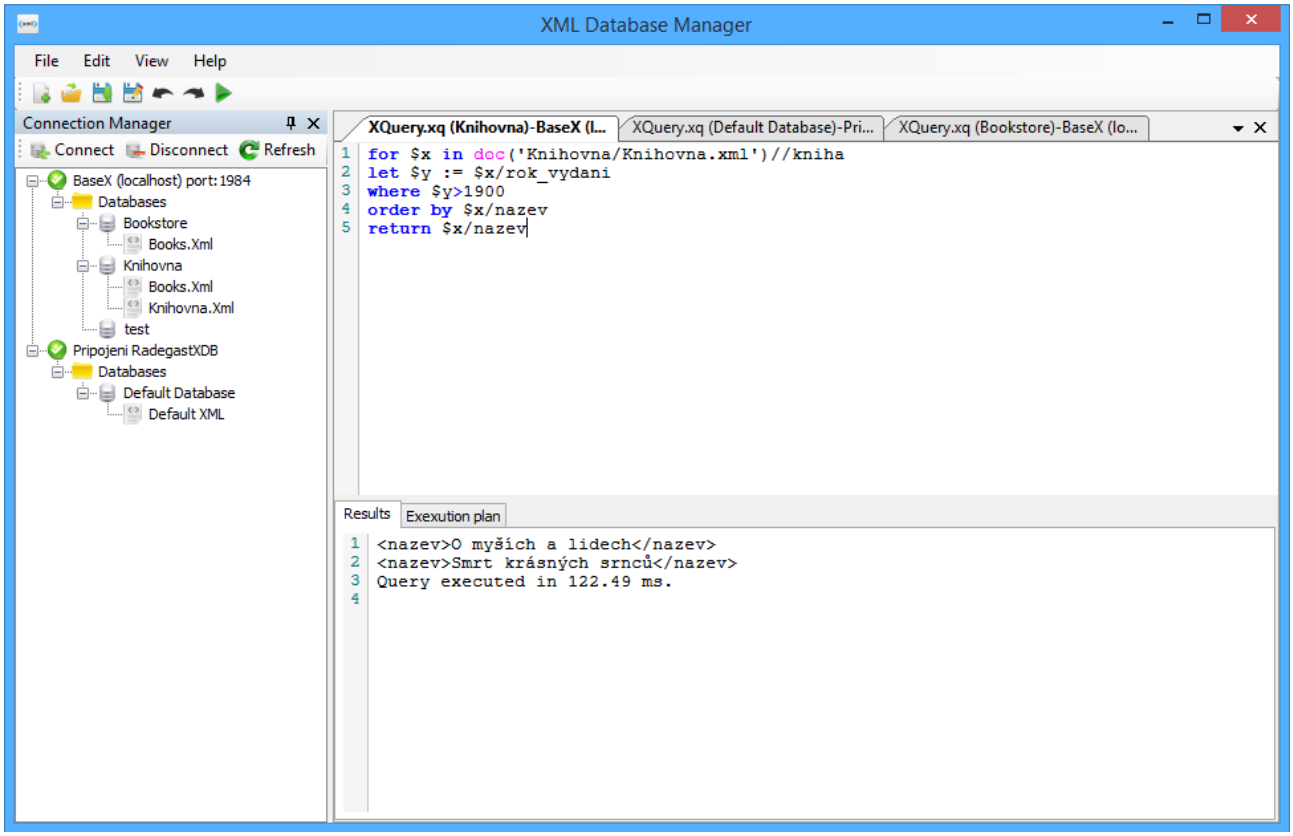
showDatabases(TreeNode node) vykresluje stromovou strukturu v Connection Manageru. Vstupním parametrem je Uzel stromu (objekt `TreeNode`), do kterého se má naplnit databáze a XML dokument.

Metody `createDatabase()`, `deleteDatabase()`, `renameDatabase()`, `databaseInfo()`, `openDatabase()`, `addXmlFile()`, `deleteXmlFile()` a `renameXmlFile()` nejsou implementovány. Je to dáno tím, že tyto operace RadegastXDB v současné verzi prototypu nepodporuje.

4 Návrh a implementace modulu pro zadávaní dotazů XQuery a prezentaci výsledku

4.1 Úvod

Jak již bylo řečeno v úvodu modulárního návrhu architektury aplikace, vizuální inspirace byla čerpána z *Microsoft SQL Server Management Studia 2012*. Proto bylo zvoleno stejné rozmístění komponent pro připojení k databázím, zadávání dotazů a prezentaci výsledků XQuery dotazů, jak lze vidět na obrázku 7.

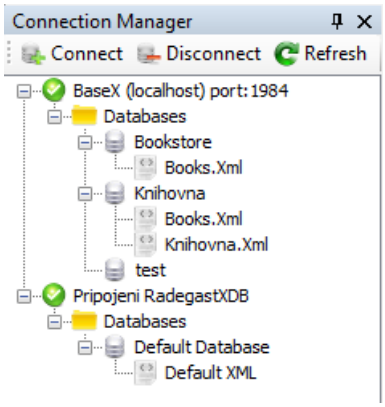


Obrázek 7: Náhled a rozložení komponent aplikace XML Database Manager

V levé části vidíme *Connection Manager*, který je nezbytný pro navázání spojení s konkrétní databází. Potom se zde nacházejí záložky formulářů pro zadávání XQuery dotazů nad určitou databází. Každý formulář ve své spodní části prezentuje vásledky XQuery dotazů v záložce *Results*, v případě dotazu nad RadegastXDB i vizualizaci plánu dotazu v záložce *Execution plan*.

4.1.1 Connection Manager

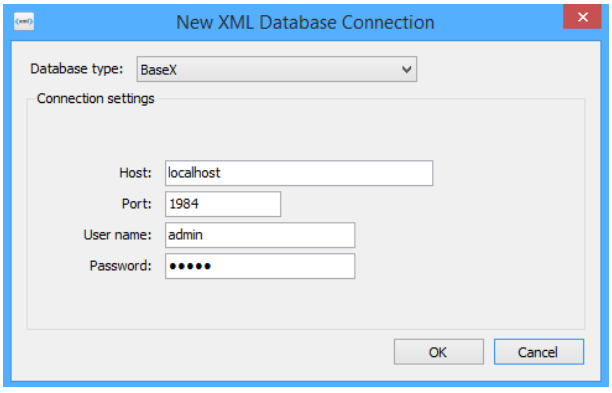
Connection manager slouží, jako správce připojení k databázím. Zobrazení jednotlivých připojení je řešeno pomocí komponenty `TreeView`. Stromová strukrura má vždy hierarchii v tomto pořadí: **popisek připojení** -> **Databases** -> **názvy databází** -> **XML soubory uložené v databázích**, viz obrázek 8.



Obrázek 8: Ukázka hierarchie v TreeView

Naplnění komponenty `TreeView` provádíme tak, že projdeme všechna aktivní připojení. Pro všechna aktivní připojení se nejprve zobrazí uzel s **popiskem připojení**, poté uzel **Databases** a následně voláme abstraktní metodu `showDatabases(TreeNode node)`, která provede naplnění uzlu podle konkrétní databáze.

V Connection Manageru je možné mít otevřeno více aktivních připojení, viz obrázek 7. Pro vytvoření nového připojení stačí kliknout na tlačítko „`Connect`“ v horní části Connection Manageru. Poté se zobrazí formulář „`New XML Database Connection`“, viz obrázek 9. Zde je možné vybrat si typ připojení `(BaseX/RadegastXDB)`, kde po vyplnění příslušných políček a po kliknutí na tlačítko „`OK`“ se vytvoří nové připojení. Pro odpojení databáze stačí kliknout na tlačítko „`Disconnect`“ a pro obnovení zobrazení stromu na tlačítko „`Refresh`“.



Obrázek 9: Ukázka formuláře pro nové připojení

4.1.2 Práce s dotazy

Pro práci s dotazy nám slouží formulář `FormQuery`. Tento formulář se skládá z části pro zadávání XQuery dotazu a části pro prezentaci výsledku. Část pro prezentaci výsledku je navíc rozdělena na záložky **Results** – zde se zobrazují výsledky dotazu a záložky **Execution plan** – zde se zobrazuje plán dotazu (viz kapitola 5).

Všechny metody pro práci s dotazy jsou naimplementované v tomto formuláři. Ve formuláři je použita komponenta `Fast Colored TextBox` 3.2.1, díky které je možné zvýraznit syntaxi XQuery dotazů. Zvýraznění syntaxe je zajištěno pomocí *regulárních výrazů*.

Zde je příklad regulárního výrazu pro zvýraznění tagu:

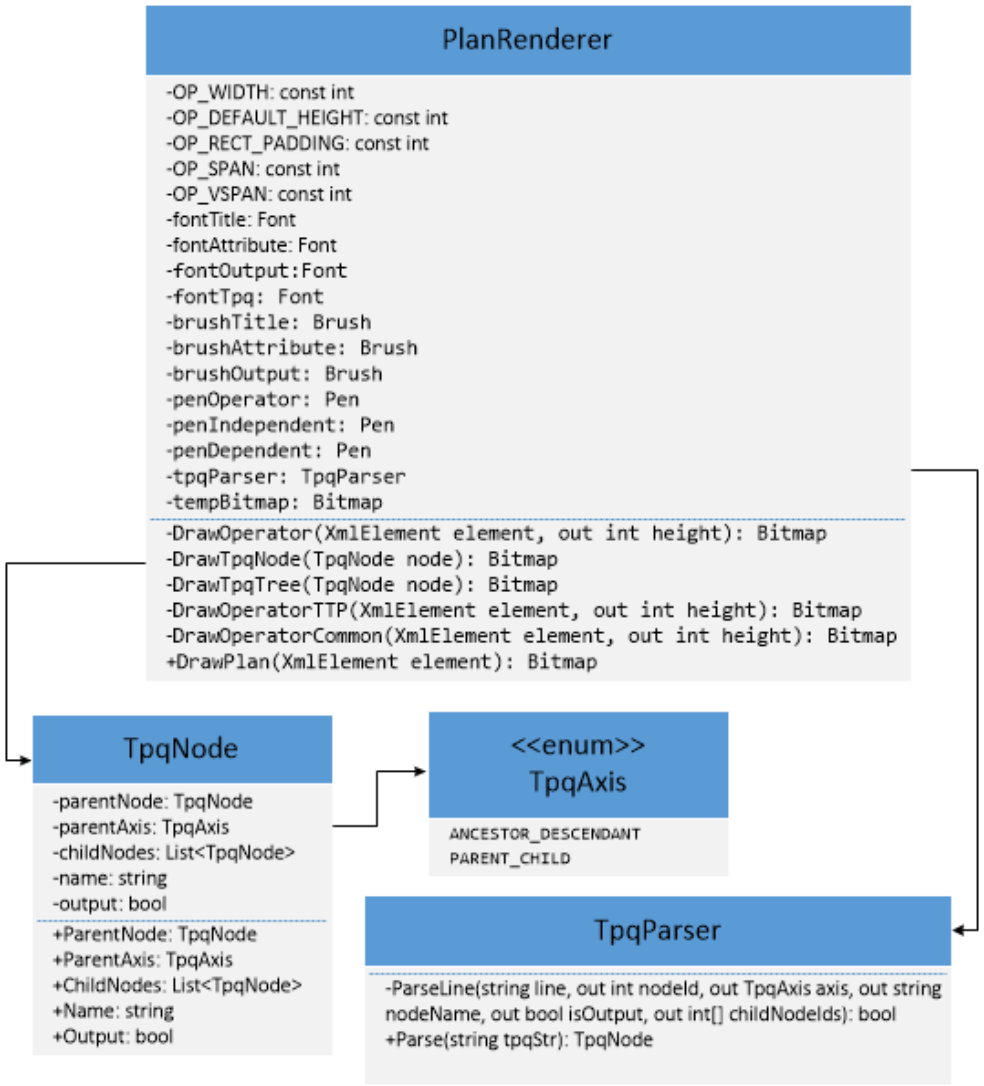
```
Regex XMLTagRegex = new Regex(@"<\?|<|/>|</|>|\?>");
```

Vykonání dotazu probíhá tím způsobem, ze uživatel klikne na tlačítko „`Run`“ v horní liště hlavního formuláře. Nejprve se zjistí pomocí metody `GetActiveForm()`, který formulář je zrovna aktivní, na něj se pak zavolá metoda `Execute()`.

V metodě `Execute()` je na základě databáze, nad kterou je dotaz volán, zavolána abstraktní metoda `ExecuteXQuery(string xquery)`, která vezme, jako vstupní parametr napsaný XQuery dotaz z části pro zadávání XQuery dotazu. V případě dotazu volaného nad databází RadegastXDB je navíc ještě jednou volána abstraktní metoda `ExecuteXQuery(string xquery)` se vstupním parametrem „`"stat('plan')"`“. Tento parametr řekne serveru, že chce vyhodnotit *plán vykonání dotazu*. Server pošle zpět plán vykonání dotazu v podobě XML, které je zpracováno `PlanRenderem` (viz další kapitola) a vykreslený plán je zobrazen v záložce „`Results`“.

5 Návrh a implementace modulu pro grafickou vizualizaci plánu dotazu

Návrh a implementace plánu dotazu byl proveden pro databázi RadegstXDB. Vykreslený plán dotazu se zobrazuje ve formuláři `FormQuery` v záložce *Execution Plan*, viz 4.1.2. Jelikož server vrací plán vykonání dotazu v podobě XML, bylo nutné toto XML zpracovat. Na třídním diagramu na obrázku 10 vidíme jednotlvé třídy používané k vykreslení plánu. V této části práce je postupně popsán význam jednotlivých tříd při vykreslování plánu.



Obrázek 10: Třídní diagram pro vykreslení plánu dotazu

5.1 Vykreslování oprátoru TupleTreePattern

Jedná se o speciální operátor zapouzdřující vyhodnocování tzv. *větvených dotazů* (`TPQ – Twig Pattern Query`). Větvený dotaz je zjednodušená podoba dotazu XPath, která specifikuje názvy hledaných XML uzlů a vztahy mezi nimi. Větvený dotaz se obvykle znázorňuje ve formě stromu – proto bylo pro operátor `TupleTreePattern` vhodné naimplementovat vlastní logiku vykreslování.

Větvený dotaz je v plánu získaném z RadegastXDB reprezentován formou textového řetězce. Ukázku textové podoby plánu vidíme na obrázku 11. Bylo tedy nutné zajistit parsování takového řetězce, rekonstrukci stromové struktury `TPQ` (třída `TpqNode` a výčet `TpqAxis`) a vykreslení `TPQ` na základě této pomocné stromové struktury.

```
<operator name="Call" time="0.015625" execute_cnt="1">
  <output />
  <attributes>
    <attribute>count</attribute>
  </attributes>
  <independent>
    <operator name="MapToItem" time="0.015625" execute_cnt="1">
      <output />
      <attributes />
      <independent>
        <operator name="TupleTreePattern" time="0.000000" execute_cnt="1">
          <output>
            <element>.ttj7</element>
          </output>
          <attributes>
            <attribute>
              0:  (1)
              ==1: &lt;-[0] (2)
              ==2: city (3)
              --3: name@.ttj7
            </attribute>
          </attributes>
```

Obrázek 11: Ukázka textové podoby plánu dotazu

5.1.1 TpqAxis

Jedná se o výčet reprezentující osu XPath `ANCESTOR_DESCENDANT` nebo `PARENT_CHILD`.

5.1.2 TpqNode

Třída reprezentující uzel `TPQ`.

5.1.3 TpqParser

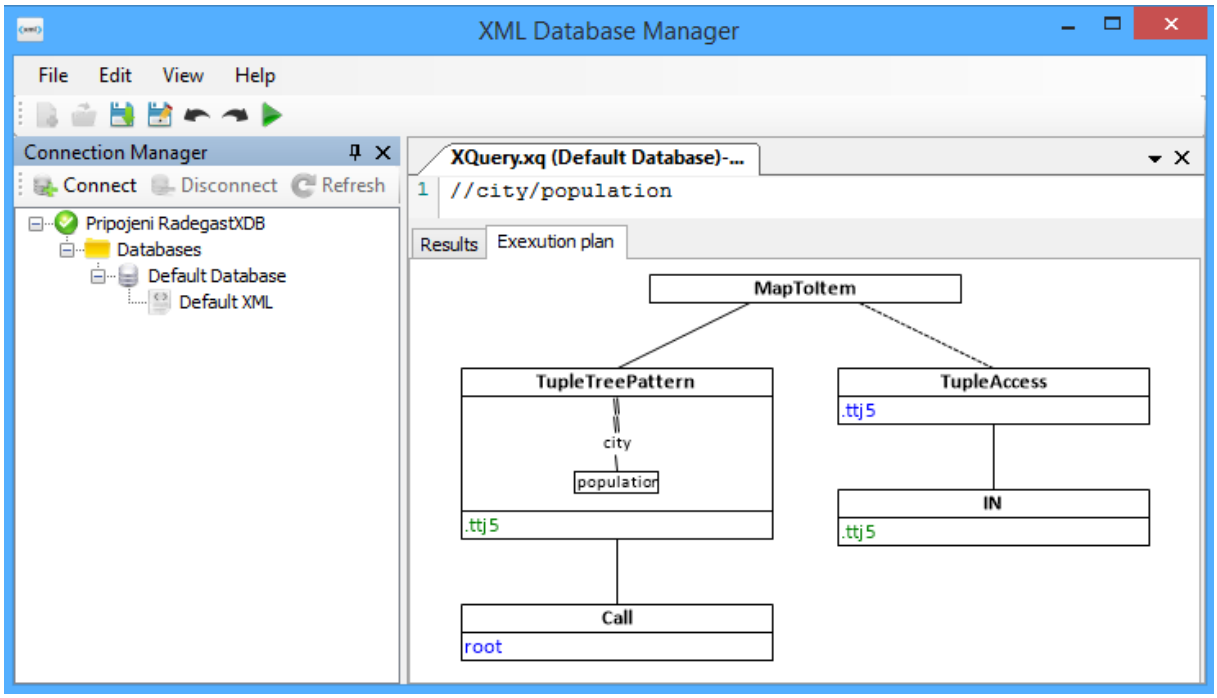
Tato třída vznikla z důvodu vykreslení atributů opetátoru `TupleTreePattern` v čitelnější podobě. Obsahuje pouze dvě metody.

Parse(string tpqStr) je metoda, která parsuje hodnoty atributů opetátoru `TupleTreePattern`. Vstupním parametrem je textový řetězec s hodnotami atributů. Návratová hodnota je `TpqNode`.

ParseLine(string line, out int nodeId, out TpqAxis axis, out string nodeName, out bool isOutput, out int[] childNodeIds) je metoda, která provádí parsování jednoho řádku atributu opetátoru `TupleTreePattern`.

5.2 PlanRenderer

Tato třída obsahuje metody pro vykreslení celého plánu dotazu. Ukázku vykreslení plánu dotazu můžete vidět na obrázku 12.



Obrázek 12: Ukázka vykreslení plánu dotazu

DrawOperator(XmlElement element, out int height) je metoda, pro kreslení operátoru. Na základě názvu operátoru, předaného jako vstupní parametr `element` se rozhodne, zda se zavolá metoda pro vyreslení obyčejného operátoru a nebo operátoru `TupleTreePattern`. Návratovou hodnotou je bitmapa.

DrawOperatorCommon(XmlElement element, out int height) je metoda, která slouží pro vykreslení obyčejného operátoru. Na obrázku 13 je vidět, jakým způsobem se počítají jednotlivé části operátoru, tj. *název*, *atributy* a *výstup operátoru*. Návratovou hodnotou je bitmapa.

Způsob vytvoření a počítání operátoru:
Nejprve je nutné vytvořit bitmapu `bitmap` s defaultní šířkou (`OP_WIDTH`) a výškou (`OP_DEFAULT_HEIGHT`). Poté vytvoříme obdélník `rectDest`, kterému nastavíme odsazení (`OP_RECT_PADDING`), šířku (`OP_WIDTH - 2 * OP_RECT_PADDING`) a výšku (`OP_DEFAULT_HEIGHT - 2 * OP_RECT_PADDING`). V takto připraveném základu můžeme začít kreslit obdélník pro název operátoru.

Vytvoříme nový obdélník `rectTitle`, začínající na počátečních hodnotách X,Y `rectDest` s šířkou stejnou, jako `recDest` a výškou `rectTitle.Bottom`, hodnotu `rectTitle.Bottom` uložíme do `yOffset`. Vykreslíme čáru.

Poté kreslíme obdélník pro atributy operátoru. Vytvoříme nový obdélník `rectAttribute`, začínající na počátečních hodnotách X (`rectDest.Left`),Y (`yOffset`) s šířkou stejnou, jako `recDest` a výškou `rectAttribute.Bottom`, hodnotu `rectAttribute.Bottom` uložíme do `yOffset`. Vykreslíme čáru.

Poté kreslíme obdélník pro výstup operátoru. Vytvoříme nový obdélník `rectOutput`, začínající na počátečních hodnotách X (`rectDest.Left`),Y (`yOffset`) s šířkou stejnou, jako `recDest` a výškou `rectOutput.Bottom`, hodnotu `rectOutput.Bottom` uložíme do `yOffset`.

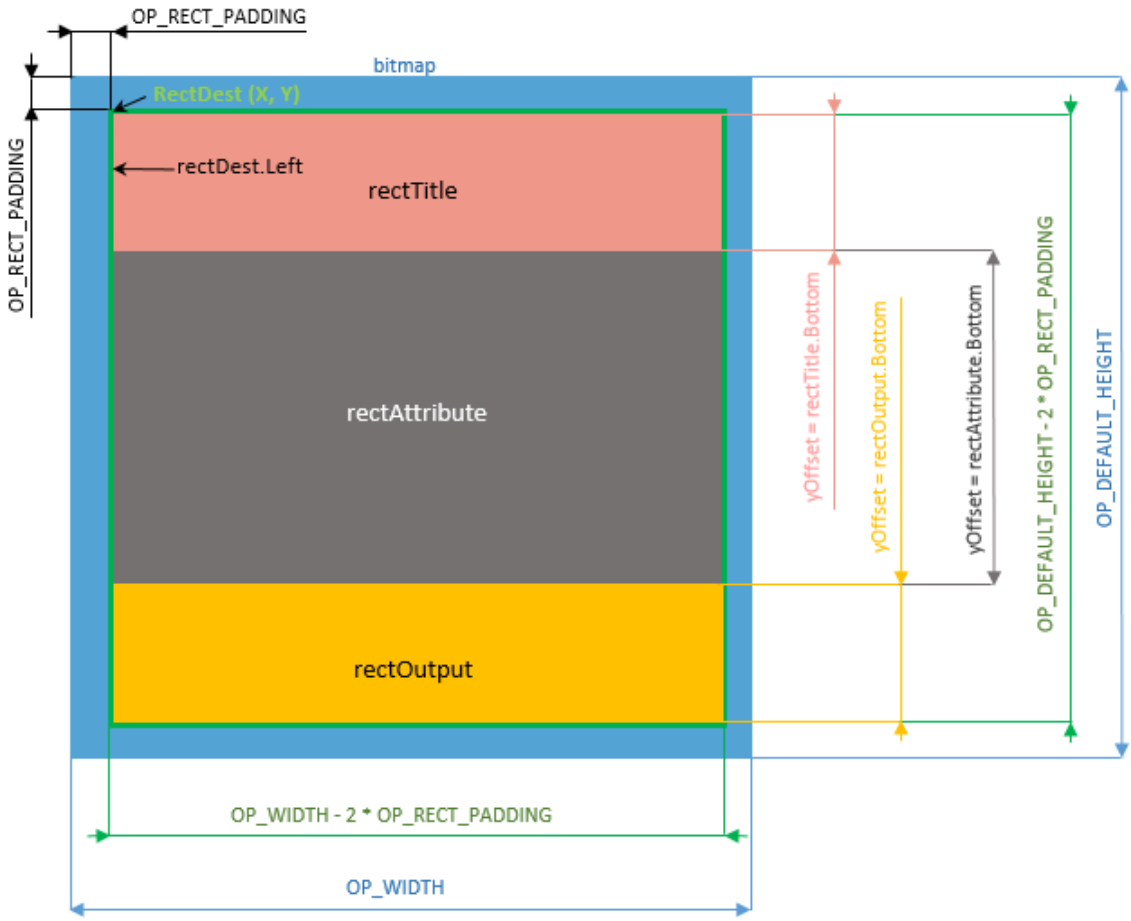
Nakonec vytvoříme obdélník `rectBorder`, kterým celý operátor orámujeme.

DrawOperatorTTP(XmlElement element, out int height) je metoda, která slouží pro vykreslení speciálního operátoru `TupleTreePattern`. Návratovou hodnotou je bitmapa.

DrawTpqNode(TpqNode node) je metoda, pomocí které se vykresluje jeden uzel atributu operátoru `TupleTreePattern`. Návratovou hodnotou je bitmapa.

DrawTpqTree(TpqNode node) je metoda, pomocí které se vykresluje celý strom atributů operátoru `TupleTreePattern`. Návratovou hodnotou je bitmapa.

DrawPlan(XmlElement element) je metoda, pomocí které se vykresluje plán vykonání XQuery dotazu.



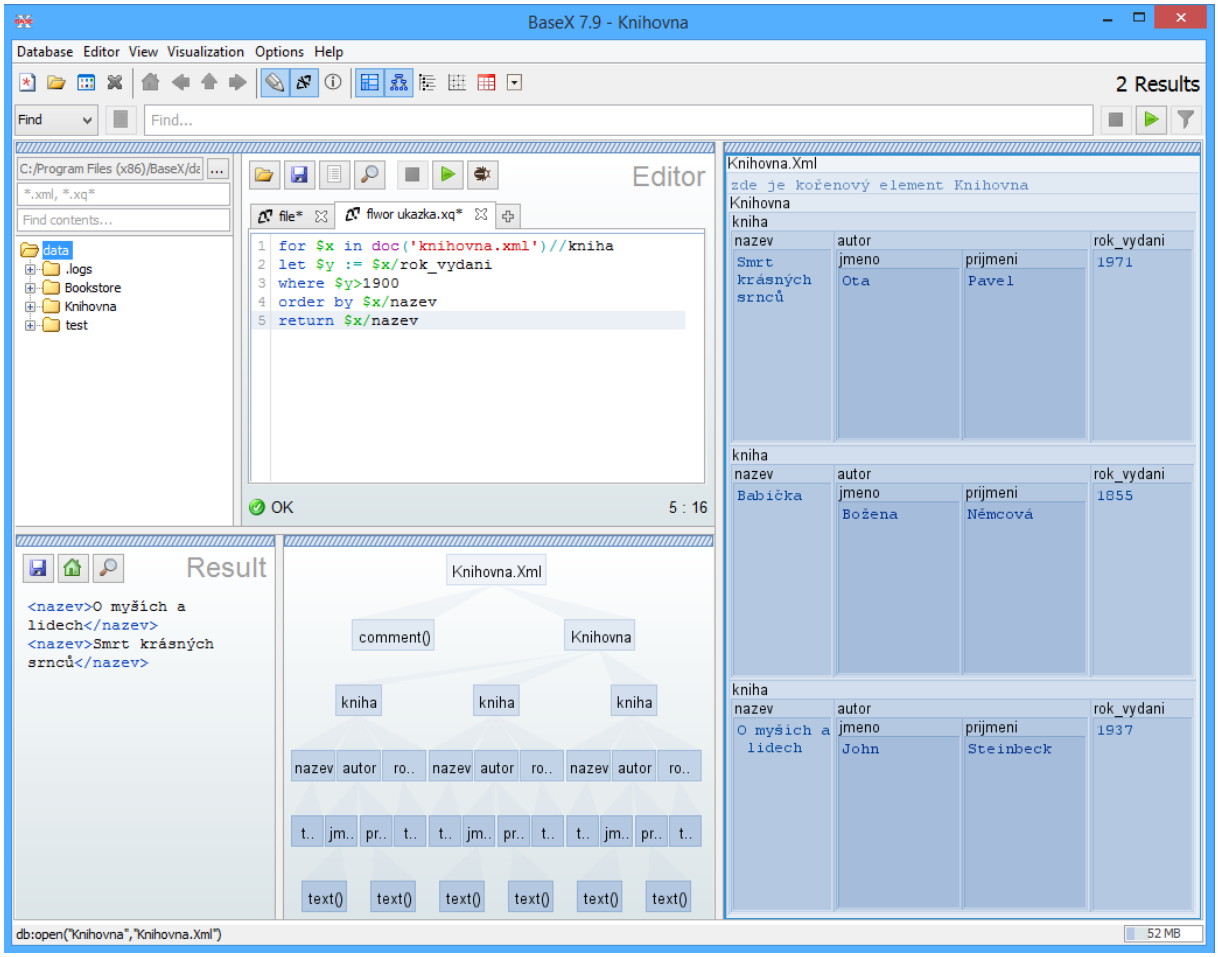
Obrázek 13: Ukázka způsobu počítání jednotlivých částí operátoru

6 Srovnání s existujícími aplikacemi

V tého části práce jsou popsány aplikace, určené ke správě nativních XML databází. Byly vybány aplikace **BaseX GUI** a **Stylus Studio X15**.

6.1 BaseX GUI

BaseX GUI je editor pro správu nativní XML databáze BaseX, viz 2.3.1. Prostředí tohoto editoru je vidět na obrázku 14.



Obrázek 14: Prostředí BaseX GUI

Na obrázku 14 je vidět rozložení jednotlivých komponent. V levé části se nachází **Project Manager**, ve které jsou soubory zobrazeny v podobe `TreeView`. Ve složce „`data`“ lze vidět aktuální databáze.

Komponenta `Editor` má standatdní funkce, jako jsou *otevření* a *uložení* souboru, dále je k dispozici funkce *Recently opened files* (umožňuje zobrazit a otevřít nedávno otevřené soubory) , funkce *find and replace* (najdi a nahraď), funkce *Stop* a *Run query*. Editor automaticky

čísluje řádky a zvýrazňuje syntaxi dotazu. Ve spodní části je zobrazena validace dotazu. Pokud je dotaz napsaný správně, zobrazí se „OK“, jinak se zobrazí chybová hláška.

Výsledky dotazů je možné zobrazit v komponentě `Result`, kde jsou možnosti *uložení* výsledku dotzu, funkce *Go home* (zobrazuje celý XML soubor) a funkce *Find* umožňuje vyhledávání textového řetězce ve výsledku dotazu. Při výpisu výsledku je zvýrazněna XML syntaxe.

Další vymožeností BaseX je možnost zobrazení komponenty `Tree`, která je zobrazena na obrázku vpravo od komponenty `Result`. Tato komponenta umožňuje zobrazit celý strom XML souboru a po kliknutí na jednotlivé uzly se v komponentě `Result` uzly zobrazí.

Nakonec je zde komponenta `Map` umístěna v právé části, která zobrazuje celý obsah XML dokumentu včetně hodnot jednotlivých uzlů. Po kliknutí na jednotlivé uzly se v komponentě `Result` zobrazí výsledek podobně, jako pomocí komponenty `Tree`.

6.2 Stylus Studio X15

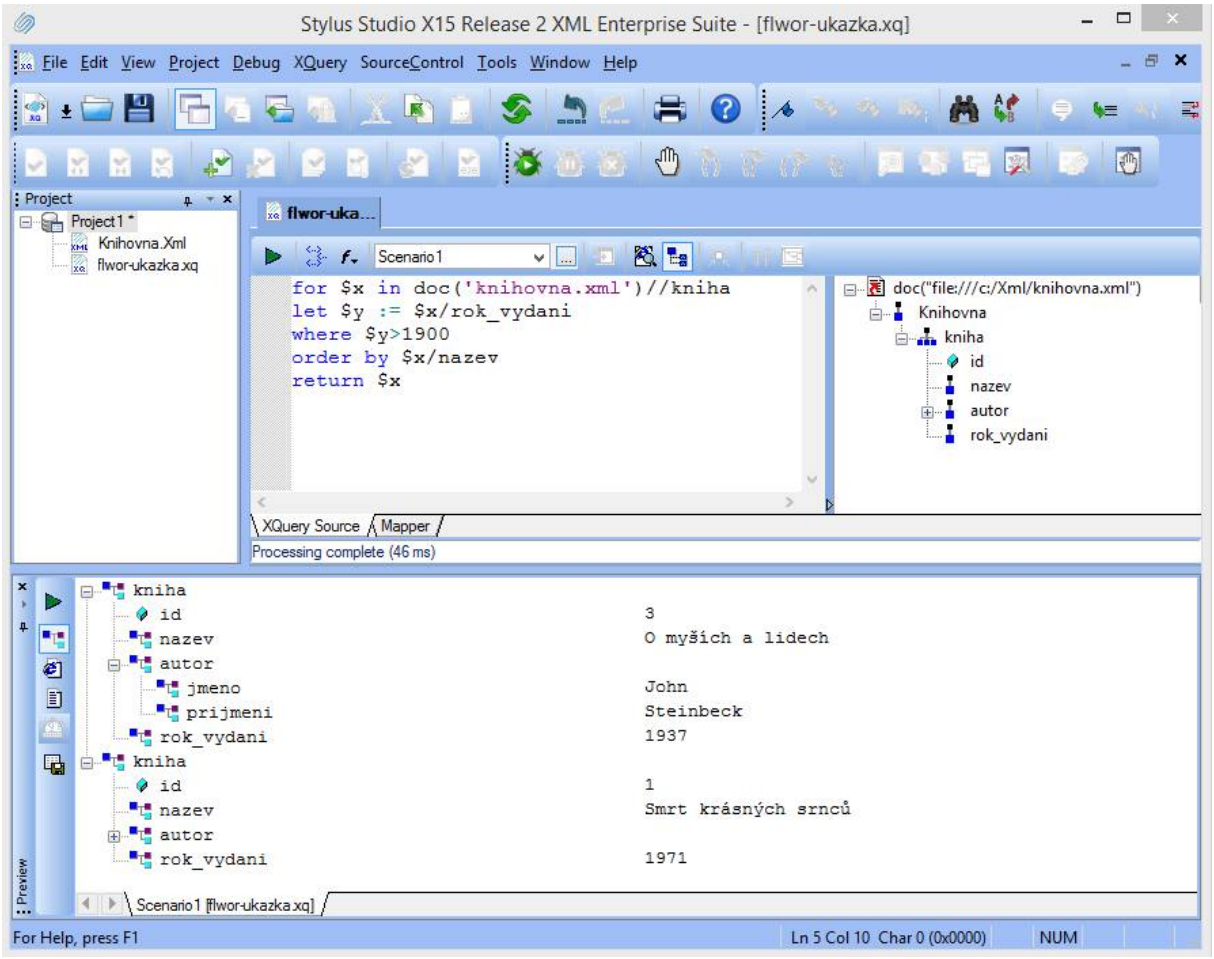
Stylus Studio X15[17] je profesionální komerční XML editor, který v sobě má celou řadu nástrojů pro práci s XML, XQuery, XPath, XHTML atd. Je zde podpora XQuery procesorů, jako jsou `Saxon`, `Data Direct XQuety` a `TigerLogic XDMS`. Samozřejmostí jsou běžné základní funkce běžného editoru.

Na obrázku 15 je vidět rozložení jednotlivých komponent. V levé části se nachází `Project Manager`, ve které jsou soubory zobrazeny v podobe `TreeView`. Zde je možné přidávat XML soubory a také soubory s XQuery dotazy.

Vpravo od `Project Manageru` je umístěna komponenta pro zadávání dotazů. Je zde zapnuté automatické zvýrazňování syntaxe a taky funkce nápovědy – při psaní dotazu se zobrazuje nabídka s možnostmi (funkce, elementy, proměnné, atd.). Mezi další funkce patří možnost zapnutí `Source Tree`, což zobrazí stromovou strukturu XML dokumentu, se kterým pracujeme. Tato stromová struktura se zobrazí vpravo. Výběrem scénáře můžeme defaultně nastavit, nad kterým XML dokumentem chceme provádět dotazy a který XQuery procesor se má použít. Možnost debuggování dotazu a validace jsou samozřejmostí.

Ve spodní části se nachází komponenta pro výpis výsledku dotazu. Výpis může být proveden formou stromu, jak lze vidět na obrázku 15 a nebo textově. Při textovém výpisu je zvýrazněna XML syntaxe.

[17] Domovská stránka Stylus Studia X15 – `http://www.stylusstudio.com/`



Obrázek 15: Prostředí Stylus Studio X15

7 Závěr

Výsledkem bakalářské práce je funkční aplikace s grafickým uživatelským rozhraním pro správu nativní XML databáze (XML Database Manager), která komunikuje s XQuery procesory BaseX a RadegastXDB a umožňuje vytvářet nad těmito databázemi XQuery dotazy, v případě RadegastXDB také vykreslení plánu dotazu.

7.1 Vlastní přínos a možnosti rozšíření

Vlastním přínosem je jednoduchá a uživatelsky příjemná aplikace, která může být v budoucnu rozšířena o možnost napojení na další XQuery procesor, například MonetDB. Jako další možností rozšíření se nabízí nápověda při psaní XQuery dotazů, kdy se uživateli během psaní zobrazí nabídka dostupných XQuery funkcí.

8 Reference

[1] W3C. *Extensible Markup Language (XML) 1.0 (Fifth Edition)* [online]. 2008 [cit. 2015-04-20]. Dostupné z: http://www.w3.org/TR/xml/

[2] W3C. *XQuery 1.0: An XML Query Language (Second Edition)* [online]. 2010 [cit. 2015-04-20]. Dostupné z: http://www.w3.org/TR/xquery/

[3] W3C. *XML Path Language (XPath) 2.0 (Second Edition)* [online]. 2010 [cit. 2015-04-20]. Dostupné z: http://www.w3.org/TR/xpath20/

[4] KOSEK, Jiří. *XML pro každého: podrobný průvodce* [online]. 1. vyd. Praha: Grada, 2000, 163 s. ISBN 80-7169-860-1. Dostupné z: http://www.kosek.cz/xml/xmlprokazdeho.pdf

[5] KOSEK, Jiří. *XQuery* [online]. Praha, 2005 [cit. 2015-04-20]. Dostupné z: http://www.kosek.cz/xml/2005devcon/xquery.pdf

[6] KOSEK, Jiří. *Základy jazyka XML* [online]. 2000-2001 [cit. 2015-04-20]. Dostupné z: http://www.kosek.cz/clanky/swn-xml/syntaxe.html

[7] LUKÁŠ, Petr. *XQuery algebra* [online]. Ostrava, 2012 [cit. 2015-04-20]. Diplimová práce. VŠB – Technická univerzita Ostrava. Dostupné z: http://hdl.handle.net/10084/93033

[8] MLÝNKOVÁ, Irena. *XML Schema a jeho implementace v prostředí relační databáze* [online]. Praha, 2003 [cit. 2015-04-20]. Diplimová práce. Univerzita Karlova v Praze. Dostupné z: http://www.ksi.mff.cuni.cz/~holubova/doc/dip2003.pdf

[9] VALČÍK, Jakub. *Editor XML dat* [online]. Praha, 2007 [cit. 2015-04-20]. Bakalářská práce. Univerzita Karlova v Praze. Dostupné z: http://www.ksi.mff.cuni.cz/~holubova/bp/Valcik.pdf

[10] BACHUREK, Ondřej. *NÁVRH VYUŽITÍ XML KOMUNIKACE V INFORMAČNÍM SYSTÉMU FIRMY* [online]. Brno, 2011 [cit. 2015-04-20]. Bakalářská práce. Vysoké učení technické v Brně. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=42618

[11] KOZEL, Jiří. *VYUŽITÍ FORMÁTU SVG PRO WEBOVOU KARTOGRAFII* [online]. Brno, 2006 [cit. 2015-04-20]. Bakalářská práce. Masarykova univerzita. Dostupné z: http://is.muni.cz/th/52087/fi_b

[12] MAREŠ, Vladimír. *Dotazovací jazyky pro XML a nativní XML databáze* [online]. České Budějovice, 2005 [cit. 2015-04-20]. Bakalářská práce. Pedagogická fakulta Jihočeské univerzity. Dostupné z: http://www.petrpexa.cz/diplomky/mares.pdf

[13] ALBRECHTOVÁ, Zdeňka. *Ukládání geodat do XML nativních databází* [online]. Plzeň, 2007 [cit. 2015-04-20]. Diplimová práce. Západočeská univerzita v Plzni. Dostupné z: http://www.gis.zcu.cz/studium/dp/2007/Albrechtova__Ukladani_geodat_do_XML_nativnich_databazi__DP.pdf

[14] RICHTA, Karel. *Jazyky XQuery a XPath* [online]. Praha, 2006 [cit. 2015-04-20]. Dostupné z: https://www.ksi.mff.cuni.cz/~richta/publications/RichtaMD2006.pdf

[15] BLAŽEK, Michal. *XML pro začátečníky - 1. část* [online]. 2007 [cit. 2015-04-20]. Dostupné z: http://programujte.com/clanek/2007030501-xml-pro-zacatecniky-1-cast/

[16] ŽIŽKA, Petr. *Nativní XML databáze – nástin teorie* [online]. 2003 [cit. 2015-04-20]. Dostupné z: https://www.interval.cz/clanky/nativni-xml-databaze-nastin-teorie/

[17] BaseX. *BaseX*. 2014. Dostupné z: http://basex.org/

[18] eXistdb. *eXistdb*. 2014. Dostupné z: http://www.exist-db.org/exist/apps/homepage/index.html

[19] MarkLogic. *MarkLogic*. 2015. Dostupné z: http://www.marklogic.com/

[20] Qualcomm. *Qualcomm® Qizx™*. 2015. Dostupné z: https://www.qualcomm.com/qizx

[21] MonetDB. *MonetDB*. 2008-2015. Dostupné z: https://www.monetdb.org/XQuery

[22] WIKIPEDIA. *eXist*. 2015. Dostupné z: http://en.wikipedia.org/wiki/EXist

[23] WIKIPEDIA. *MarkLogic*. 2015. Dostupné z: http://en.wikipedia.org/wiki/MarkLogic#Licensing_and_Support

[24] KRÁTKÝ, Michal, DVORSKÝ, Jiří. *Úvod do programování* [online]. Ostrava, 2004-2005. Dostupné z: http://www.cs.vsb.cz/kratky/courses/2004-05/udp/presentation/udp-10_6.pdf

9 Přílohy

Součástí bakalářské práce je přiložené CD obsahující zdrojové kódy aplikace, elektronickou verzi tohoto dokumentu, návod na použití aplikace, instalační soubor databáze BaseX, spustitelný RadegastXDB server a testovací XML soubor.

Obsahuje adresáře:

- BaseX
  - instalační soubor BaseX79.exe
- bin
  - spustitelný soubor XML_Database_Manager.exe
- doc
  - elektronická verze tohoto dokumentu
  - návod na použití aplikace
- RadegastXDB
  - bin
    - spustitelný RadegastXDB server xdb.exe
  - settings
    - konfigurační soubor settings.xml
  - index
  - testovací XML soubor
- src
  - projekt ve Visual Studiu 2012, zdrojové soubory mají příponu *.cs